Data Compression Techniques For Maps

M.Y. Jaisimha, H. Potlapalli
H. Barad, A.B. Martinez
Electrical Engineering
Tulane University
New Orleans, Louisiana

M.C. Lohrenz, J. Ryan
Naval Ocean Research & Development Activity
J.C. Stennis Space Center, Mississippi

J. Pollard
Planning Systems, Inc.
Slidell, Louisiana

ABSTRACT

The efficiencies of various data compression techniques as applied to color maps are compared. These color maps have certain special characteristics such as large homogeneous regions and fine detail such as lines and lettering. The color maps are first classified using the K means clustering algorithm with neighborhood classification. Three techniques are investigated - contour, quadtree and run-length coding. The run-length coding algorithm is modified to allow wrap around of runs. A modification of the standard binary image quadtree compression algorithm for color images is introduced. In quadtree coding a modified eldest-son eldest younger sibling quadtree is used to reduce memory requirement in storing the quadtree. Lempel-Ziv compression is applied to the classified and unclassified images as also to the output of the compression algorithms. The algorithms will be compared on the compression ratios achieved.

INTRODUCTION

This paper presents a comparative evaluation of various compression techniques as applied to color maps. The maps have certain specific characteristics such as large homogeneous regions and fine detail. The compression techniques being considered, deliver good compression only when the images have large homogeneous regions. The maps are classified using a neighborhood classification algorithm before the compression algorithms are applied to the images. Run-length, contour and quadtree coding are investigated.

The maps in question are digitized versions of printed maps. Due to the nature of the printing of the maps, the images end up having a grainy salt and pepper character with each pixel a different color. The digitization process introduces some inhomogeneity in the coloring because of the printing, which involves half-toning. Further, the maps have map to map variation in coloring due to the varying ages and printing techniques of the maps. The result of all this is that the maps do not have uniform coloring. The maps under consideration also have fine detail such as latitude and longitude lines and lettering. These characteristics place a bound on the degree of compression that can be achieved with the compression techniques.

The maps are classified to overcome the limitation due to inhomogeneity in the maps. The K means clustering algorithm was used to classify the image into a much smaller number of classes (8 as compared to the 256 colors present in the original images)[1,3]. The maps are also classified using vector quantization with differing number of codewords and various distance criteria[6,8]. Neighborhood classification was applied in order to improve the quality of the image.

The presence of large homogeneous regions in the maps suggests that techniques such as quadtree and run-length coding would be applicable. Quadtree compression has been applied to binary images and has been shown to give good compression[11]. In [15] the authors have applied quadtree compression to the compression of line drawing map overlays. A 512 x 512 image could be stored in as few as 130 nodes. A modified version of the standard quadtree algorithm, applicable to color images is proposed and evaluated. Run-length and contour coding have been the subject of extensive research. The run-length coding algorithm of [2] and contour coding algorithm of [16] are applied to color map compression. Since the images have large homogeneous regions, the runs are allowed to wrap around from one row to another. Before applying the compression algorithms the images are converted from 24 bits/pixel to 8 bits/pixel pseudo-color images. The image files now consist of a gray level image file and a color look up table. Thus compression of 3:1 is already achieved before any compression techniques are applied.

COMPRESSION TECHNIQUES

Quadtree Coding

Quadtree coding as applied to color maps is discussed in the following section. Quadtree compression for binary images has been the subject of substantial research[11]. The presence of large homogeneous regions in the maps, together with the small number of colors led to the choice of quadtree compression. A variant of the standard quadtree algorithm as applied to binary images is used. The rooted tree approach to a quadtree is used to minimize the memory requirement in storing the quadtree.

The efficiency of quadtrees in image processing applications is based on the principle of recursive decomposition[11]. Recursive decomposition of a picture is performed by successively decomposing the image into smaller and smaller regular polygons. These regular polygons or tilings of the image plane are referred to as tessellations. Typical tessellations are squares, equilateral triangles, hexagons and isosceles right triangles. Due to the presence of large homogeneous regions in the image, the feasibility of using

CH2674-5/89/0000-0878$01.00©1989IEEE

Report Documentation Page

Form Approved
OMB No. 0704-0188

1. REPORT DATE **1989**	2. REPORT TYPE	3. DATES COVERED **00-00-1989 to 00-00-1989**
4. TITLE AND SUBTITLE **Data Compression Techniques for Maps**		5a. CONTRACT NUMBER 5b. GRANT NUMBER 5c. PROGRAM ELEMENT NUMBER
6. AUTHOR(S)		5d. PROJECT NUMBER 5e. TASK NUMBER 5f. WORK UNIT NUMBER
7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES) **Naval Ocean Research & Development Activity,Stennis Space Center ,MS,39529**		8. PERFORMING ORGANIZATION REPORT NUMBER
9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)		10. SPONSOR/MONITOR'S ACRONYM(S) 11. SPONSOR/MONITOR'S REPORT NUMBER(S)

12. DISTRIBUTION/AVAILABILITY STATEMENT
Approved for public release; distribution unlimited

13. SUPPLEMENTARY NOTES
Proceedings of the IEEE: Energy and Information Technologies in the Southeast, Vol.2, pp. 878-883. U. of SC, Swearingen Engineering Center, Columbia, SC. April

14. ABSTRACT
see report

15. SUBJECT TERMS

16. SECURITY CLASSIFICATION OF:			17. LIMITATION OF ABSTRACT	18. NUMBER OF PAGES	19a. NAME OF RESPONSIBLE PERSON
a. REPORT **unclassified**	b. ABSTRACT **unclassified**	c. THIS PAGE **unclassified**	**Same as Report (SAR)**	**6**	

Standard Form 298 (Rev. 8-98)
Prescribed by ANSI Std Z39-18

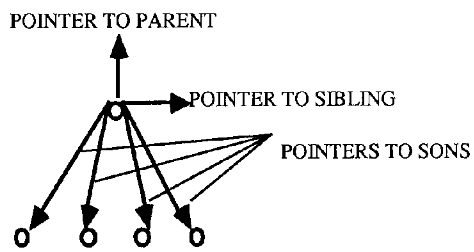

Figure 1. Conventional quadtree representation.

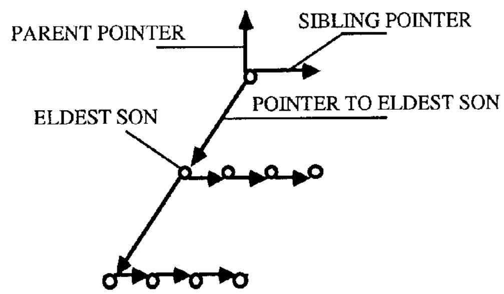

Figure 2. Eldest son-eldest younger sibling quadtree.

octrees instead of quadtrees was explored. Since octagonal tessellations are not unlimited, this is not feasible and square tessellations were chosen.

The quadtree can be considered as a variant of a binary tree with each parent having four sons (Figure 1). Typically, each node of a quadtree consists of a value element, pointers to the 4 children, a pointer to the parent and a pointer to the sibling – a total of 6 pointers plus one value element per node. The amount of memory used to set up the quadtree is reduced by using only three pointers per node. The structure used is called a rooted tree [7,17]. We store a rooted tree using the eldest child - eldest younger sibling method illustrated in Figure 2. Each node of the tree consists of one value (color) element, one son pointer which points to the eldest of the nodes four children, a sibling pointer which points to the next youngest sibling and a parent pointer which points to the parent of the node. Pointers are set to NIL if no nodes exist to which they are to point.

In addition modifications have also to be made to the binary image algorithm, to account for the larger number of colors in the image. The classified image now has one of eight possible colors. The colors are of two types : pure colors, which occur in the image and a shade called GRAY. The shade GRAY is used in the quadtree compression algorithm to indicate that a particular node has sons of differing pure colors.

In order to permit ease in computation and reconstruction of the image a numbering scheme for the nodes of the quadtree is introduced. The root node has a number 0. Further levels are divided into blocks ok 2 x 2 and then numbered in a clockwise ascending order. This numbering scheme is illustrated in Figure 3. Also the input image had a size of 575 x 639. This odd number of elements would result in a major handling overhead. The quadtree method works best when the size of the image is of the form $2^n \times 2^n$, where n is an integer. The bottom most level of the quadtree was assumed to be of size $2^{10} \times 2^{10}$. The image was then read into the top left hand corner of the bottom level.

Operation of quadtree compression is explained below:

set up tree()

if (they are of the same color)
then
  color of parent = color of childrenif(four blocks are the same color and the color is not GRAY)
then
  son pointer of parent is set to NIL
if (the block under consideration is the root of the tree)
then
  stop

Once we reach the root of the tree, the tree is trimmed - since a parent value equal to a pure color (not GRAY) represents an entire block of pixels in the image which possess this particular color. Thus it would make sense to store only the parent and trim off the succeeding generations of children. Since the images possess large homogeneous regions some of the lower generations of the tree can be discarded. In this manner it is possible to store the entire image with only a portion of the quadtree. This results in memory savings.

1	2	5	6
4	3	8	7
9	10	13	14
12	11	16	15

numbers run from 0 thru 1398100

Figure 3. Numbering scheme for quadtree nodes.

A modified quadtree traversal algorithm for the rooted quadtree is given below. In [14] a quadtree traversal algorithm is described for a quadtree with each node having four son pointers and a parent pointer. The three pointer configuration of the nodes being used necessitate a different traversal algorithm. The algorithm used was a modified version of the standard preorder traversal algorithm [7].The large number of recursive calls to the tree traversal procedure could result in stack overflow. To avoid this the tree traversal algorithm is applied separately to each of the four sub-trees of the root.

The traversal algorithm is described below in pseudo-code

```
traverse(node)
if (node has no children)
then
    visit(node)
    if (node has a sibling)
    then
        traverse(sibling)
    if (node has no sibling)
    then
        if(parent of node has sibling)
        then
            traverse(sibling of the parent of the
                          node)
        else
            traverse(sibling of the "grand"
                          parent of the node)
if(node is a root)
then
    visit(node)
    traverse(son)
return(when node is root of the quadtree)
```

In visit(node) the node number and value of each element was written to an output file.

The decoding algorithm involves setting up the quadtree as a first step. The output file of the quadtree compression algorithm consists of colors of nodes and node numbers of the portion of the quadtree that has to be retained. These are then allotted to the appropriate nodes.

The decoding algorithm for quadtree compression is described below

```
decode quadtree()
for each level of the quadtree
if(level is not the bottom level)
then
    for each node in the level
    if (color of node is a pure color)
    then
        children of node have same color as
          parent
if (level is bottom level)
then
    stop
```

Run-length Coding

The presence of large homogeneous regions in the maps indicates that run-length coding would result in good compression. The presence of fine detail and lettering on the maps results in a limit on the length of the runs and hence on the compression attainable. Since large regions of the images have the same color, runs are allowed to wrap around from one row of pixels to another.

Run length coding exploits length wise correlations that exist in the maps. The coding algorithm described in [2] is used. The operation of the run length coding algorithm is summarized below

```
run length coding(row)
for each pixel
if(next pixel in the same row is of the same color)
then
    length of run = length of run + 1
if(next pixel in the same row is not the same
color)
then
    store length of run and color
if(end of row is reached)
then
    continue run on row + 1
```

In the maps under consideration, there are many long runs. Substantial memory savings result from the fact that only the color of the first pixel in a run and the length of the run have to be stored. The unclassified image is a salt and pepper image and hence results in poor compression. Classification of the images improves the performance of the algorithm.

Contour Coding

The maps have large regions of only one color. Contour coding achieves memory savings by storing the the perimeter and the color of each region. The algorithm used for contour coding is the one given by Wilkins in[16].

Each contour is uniquely determined by the color of the initial point of the contour, its location, and a sequence of directionals that give the direction of travel around the contour. The contour coding algorithm used consists of two parts : the T algorithm that traces the contour and the IP algorithm which determines whether a pixel is the initial point of a new contour.

The T algorithm traces contours based on the LML (Left Most Looking) rule which is described below:

```
LML rule(contour)
for each point
if(pixel on left has same color)
then
    move left
else
    if(pixel ahead has same color)
    then
        move ahead
    else
        if(pixel to the right has same color)
        then
            move right
        else
            move back
if(new point is the initial point)
then
    IP algorithm()
else
    continue LML(contour)
```

Indicators are assigned to each point to distinguish initial points from points on the interior of a contour. All pixels are initially assigned an initial indicator. The indicator is then reassigned based on the direction of travel into and out of the pixel. The IP algorithm begins at the next pixel to the right of the initial point of the last contour and then scans it row by row for the next initial point. The IP algorithm which locates initial points of contours is described on the following page.

```
IP algorithm()
if(indicator is not the same as initial indicator)
then
    next pixel
if(indicator is the initial indicator and color same as
previous initial point)
then
    next pixel
else
    pixel is initial point
```

RESULTS

The unclassified and K means classified maps were compressed using Lempel-Ziv coding. The unclassified image gave a compression ratio of only 1.4:1. The K means classified image could be compressed by a ratio of 8:1. A typical image is compressed using run-length coding. The compression ratio obtained is 2.8:1. The total compression time is 12 seconds and the decoding time is 9 seconds. The compression achieved here is expected because of the presence of large homogeneous regions. The length of the longest run is 520 pixels. The output of the run-length coder for the classified image was Lempel-Ziv coded resulting in a compression of 6.7:1. A histogram of the lengths of the runs is presented in Figure 4.

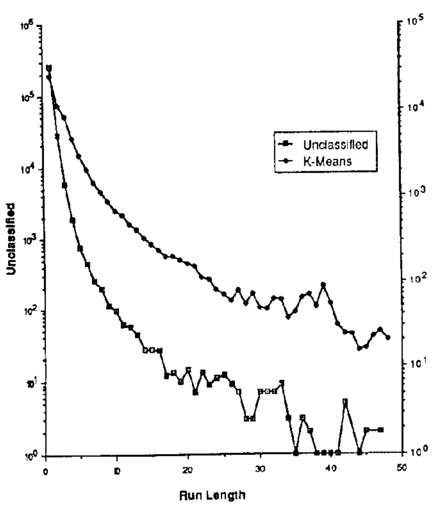

Figure 4. Histogram of run lengths.

Quadtree coding of the maps resulted in a compression of 7:1. The output of the quadtree coding algorithm was then compressed using Lempel-Ziv coding. The compression ratio achieved was 10.5:1. The coding and decoding times for quadtree coding is of the order of 10 minutes. All compression ratios mentioned above are effectively multiplied by a factor of 3 when the initial transformation from a 24 bits/pixel image to a pseudo-color image, is taken into consideration.

CONCLUSIONS

The exponential behavior of the histogram of the runs indicates that runs of short run length have higher probability. Accordingly Huffman coding of the runs would result in more efficient bit assignment and hence greater compression ratios. The improvement in overall compression ratio when Lempel-Ziv coding is applied to the output of the run-lenght coder futher indicates that Huffman coding of the runs would improve the compression ratio. The quadtree coding approach offers a higher compression ratio than run-length coding. The comparison of computation times showed that run-length coding is significantly faster than quadtree coding. Further improvements will have to be made in quadtree coding to decrease computation time and memory requirement for the decoding algorithm. The output of the quadtree coder uses 32 bits to store the node number of each node. Further compression can be achieved by using only the smallest number of bits required to store the node numbers.

ACKNOWLEDGEMENT

This work was funded under Aircraft Procurement, Navy H1CC Subhead AV- 8B Harrier, Program Elemets 9410101(64262N) and 980101 (APN) and NORDA contract N00014 - 88 - K - 6006. Approved for public release, distribution is unlimited. NORDA contribution number PR89:012:351.

REFERENCES

[1] Richard O. Duda and Peter E. Hart, "Pattern Classification and Scene Analysis," John Wiley and Sons Inc., New York, 1973.

[2] Rafael C. Gonzalez and Paul Wintz, "Digital Image Processing," 2nd. ed., Addison-Wesley Publishing Co., Reading, Mass., 1987.

[3] John A. Hartigan, "Clustering Algorithms," John Wiley and Sons Inc., New York 1975.

[4] A.K.Jain, "Image data compression - A review," Proc. IEEE, Vol.69, pp.349-389, Mar. 1981.

[5] A.K.Jain, "Advances in Mathematical models for image processing," Proc. IEEE, Vol.69, No.5, pp. 502-528, May 1981.

[6] N.S.Jayant and P.Knoll, "Digital Coding of Waveforms," Prentice-Hall Inc.

[7] Donald E. Knuth, "The Art of Computer Programming Vol.2, Fundamental Algorithms," McGraw-Hill, 1978 .

[8] Y.Linde, A.Buzo, R.M.Gray, "An algorithm for vector quantizer design," IEEE Trans. on Comm., Vol. COM-28, pp.84-95, Jan.1980.

[9] H.Meyr, H.G.Rosdolsky and T.S.Huang, "Optimum run-length code," IEEE Trans. on Comm., Vol. COM-22, pp.826-835, June 1974.

[10] A.Rosenfeld and Avinash C. Kak, " Digital Picture Processing," 2nd ed., Academic Press Inc., New York 1982.

[11] H.Samet, "The quadtree and related data structures," ACM Computing Surveys, Vol.16, No.2, pp.188-260, June 1984.

[12] H.Samet, "Algorithms for conversion of quadtrees to rasters," Computer Vision, Graphics and Image Processing, Vol.26, pp.1-16, 1984.

[13] H.Samet, "Data structures for quadtree approximation and compression," Communications of the ACM, Vol.28, No.9, pp.973-993, Sep. 1985.

[14] H.Samet, "A top-down quadtree traversal algorithm," IEEE Trans. on Pattern Anal. and Mach. Intell., Vol. PAMI-7, No. 1, pp.94-98, Jan. 1985.

[15] H.Samet, A.Rosenfeld, Clifford. A. Shaffer and Robert E. Webber, "Quadtree region representation in cartography: Experimental results," IEEE Trans. on Systems, Man and Cybernetics, Vol. SMC-13, No.6, pp.1148-1154, Dec.1983.

[16] L.C.Wilkins, "Studies on data compression, Part I: Picture coding by contours," Ph.D. dissertation, 1970, School of Electrical Engineering, Purdue University, Lafayette, Indiana.

[17] Christopher J. Van Wyk, "Data Structures and C Programs," Addison Wesley Pub. Co., Reading, Mass., 1988.

HERB BARAD

Herb Barad has been an Assistant Professor of Electrical Engineering at Tulane University since 1988. He received his BS in Electrical Engineering and his BS in Computer Science from Tulane University in 1981. He received his MS in 1983 and his PhD in 1987, both in Electrical Engineering from the University of Southern California. He is a member of IEEE, IEEE Computer Society, Tau Beta Pi, Eta Kappa Nu, and Sigma Xi. His research interests include image processing, vision, computer graphics, parallel architectures, and rapid prototyping. He is co-director of the Signal and Image Processing Laboratory (SIMPL) at Tulane.

M.Y.JAISIMHA

M.Y.JAISIMHA has been a Graduate Student in the Dept. of Electrical Engineering at Tulane University since 1987. He received his BE in Electronics & Communication Engineering from the University of Roorkee, India in 1987. He is a Student member of IEEE & IEEE Computer Society.

H.POTLAPALLI

H.POTLAPALLI has been a Graduate Student in the Dept. of Electrical Engineering at Tulane University since 1987. He received his BE in Electrical Engineering(Instrumentation) from Osmania University , India in 1987. He is a Student member of IEEE .

ANDREW B. MARTINEZ

Andrew B. Martinez received the B.S. in electrical engineering from Tulane University, New Orleans, LA in 1978. He received the M.S. and Ph.D. in electrical engineering from Princeton University, Princeton, NJ in 1979 and 1981. Since 1982, he has been a faculty member in the Department of Electrical Engineering at Tulane University where he is currently an Assistant Professor. His research interests include statistical communications, adaptive filtering, and digital signal processing. He is a member of IEEE, Tau Beta Pi, Eta Kappa Nu. He is co-director of the Signal and Image Processing Laboratory (SIMPL) at Tulane.

James Ryan

James Ryan is an electrical engineer with the Mapping, Charting and Geodesy division of the Naval Ocean Research and Development Activity since 1986. He received his B.S. in Electrical Eng. and a B.A. in Philosophy from the University of New Orleans. He has been an Electronics Engineer with the U.S. Naval Oceanographic Office and an Associate Systems Engineer with Tano Corporation.

Maura Connor Lohrenz

Maura Lohrenz has been a Physical Scientist with the Mapping, Charting and Geodesy division of Naval Ocean Research and Development Activity since 1986. She received her B.A. in Biology from Middlebury College, CT. She has also as a research assistant with the U.S. Geographical Survey and as computer programmer/analyst with Woods Hole Oceanographic Institution.